getfield

Operation

Fetch field from object

Format

getfield
indexbyte1
indexbyte2

Forms

getfield = 180 (0xb4)

Operand Stack

..., *objectref* ⇒ ..., *value*

Description

The *objectref*, which must be of type `reference`, is popped from the operand stack. The unsigned *indexbyte1* and *indexbyte2* are used to construct an index into the runtime constant pool of the current class (§3.6), where the value of the index is (*indexbyte1* << 8) | *indexbyte2*. The runtime constant pool item at that index must be a symbolic reference to a field (§5.1), which gives the name and descriptor of the field as well as a symbolic reference to the class in which the field is to be found. The referenced field is resolved (§5.4.3.2). The *value* of the referenced field in *objectref* is fetched and pushed onto the operand stack.

The class of *objectref* must not be an array. If the field is `protected` (§4.6), and it is either a member of the current class or a member of a superclass of the current class, then the class of *objectref* must be either the current class or a subclass of the current class.

Linking Exceptions

During resolution of the symbolic reference to the field, any of the errors pertaining to field resolution documented in Section 5.4.3.2 can be thrown.

Otherwise, if the resolved field is a `static` field, *getfield* throws an `IncompatibleClassChangeError`.

Runtime Exception

Otherwise, if *objectref* is `null`, the *getfield* instruction throws a `NullPointerException`.

Notes

The *getfield* instruction cannot be used to access the `length` field of an array. The *arraylength* instruction is used instead.

---

getstatic

Operation

Get `static` field from class

Format

getstatic
indexbyte1
indexbyte2

Forms

getstatic = 178 (0xb2)

Operand Stack

..., ⇒ ..., *value*

Description

The unsigned *indexbyte1* and *indexbyte2* are used to construct an index into the runtime constant pool of the current class (§3.6), where the value of the index is (*indexbyte1* << 8) | *indexbyte2*. The runtime constant pool item at that index must be a symbolic reference to a field (§5.1), which gives the name and descriptor of the field as well as a symbolic reference to the class or interface in which the field is to be found. The referenced field is resolved (§5.4.3.2).

On successful resolution of the field, the class or interface that declared the resolved field is initialized (§5.5) if that class or interface has not already been initialized.

The *value* of the class or interface field is fetched and pushed onto the operand stack.

Linking Exceptions

During resolution of the symbolic reference to the class or interface field, any of the exceptions pertaining to field resolution documented in Section 5.4.3.2 can be thrown.

Otherwise, if the resolved field is not a `static` (class) field or an interface field, *getstatic*

throws an `IncompatibleClassChangeError`.

Runtime Exception

Otherwise, if execution of this *getstatic* instruction causes initialization of the referenced class or interface, *getstatic* may throw an `Error` as detailed in Section 2.17.5.

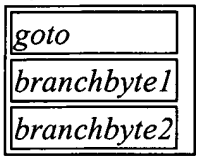
---

goto

Operation

Branch always

Format



Forms

goto = 167 (0xa7)

Operand Stack

No change

Description

The unsigned bytes *branchbyte1* and *branchbyte2* are used to construct a signed 16-bit *branchoffset*, where *branchoffset* is (*branchbyte1* << 8) | *branchbyte2*. Execution proceeds at that offset from the address of the opcode of this *goto* instruction. The target address must be that of an opcode of an instruction within the method that contains this *goto* instruction.

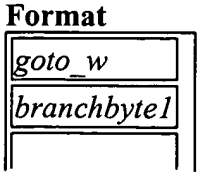
---

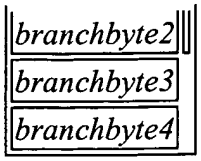
goto_w

Operation

Branch always (wide index)

Format





Forms

goto_w = 200 (0xc8)

Operand Stack

No change

Description

The unsigned bytes *branchbyte1*, *branchbyte2*, *branchbyte3*, and *branchbyte4* are used to construct a signed 32-bit *branchoffset*, where *branchoffset* is (*branchbyte1* << 24) | (*branchbyte2* << 16) | (*branchbyte3* << 8) | *branchbyte4*. Execution proceeds at that offset from the address of the opcode of this *goto_w* instruction. The target address must be that of an opcode of an instruction within the method that contains this *goto_w* instruction.

Notes

Although the *goto_w* instruction takes a 4-byte branch offset, other factors limit the size of a method to 65535 bytes (§4.10). This limit may be raised in a future release of the Java virtual machine.

Contents | Prev | Next | Index